FIT3077 - Sprint 3

Alex Ung, Romal Patel, Hirun Hettigoda

Table of Contents

Code Review

Assessment Criteria

Completeness of solution - [functional completeness as well as correctness]

Is the rationale appropriate? [functional appropriateness]

Understandability of the solution [appropriateness recognisability]

Extensibility [modifiability]

Quality of written source code (coding standards, reliance on case analysis and/or down-casts) [maintainability]

Aesthetics (how good is the UI)

Criteria	0 - Unsatisfactory	1 - Needs Improvement	2 - Satisfactory	3 - Very Good	4 - Excellent
Functional Completeness	Key functionalities are mostly absent or incorrectly implemented.	Some functionalities are present but several key aspects are flawed or missing.	Basic functionalities are present but some are missing or flawed.	Most key functionalities are covered and correctly implemented.	All key functionalities are comprehensively covered and correctly implemented.
Functional Appropriateness	Solution direction lacks clear rationale or is inappropriate for the project goals.	Solution somewhat aligns with project goals but significant adjustments needed.	Solution direction has a basic rationale that aligns somewhat with project goals.	Solution direction is well-reasoned and aligns well with project goals.	Solution direction is excellently reasoned and perfectly aligns with project goals.

Appropriateness Recognisability	Solution direction is unclear or confusing, making it hard to understand.	Solution is somewhat understandable but lacks clarity in important areas.	Solution direction is understandable but could be clearer or more intuitive.	Solution direction is clear and well-understood by stakeholders.	Solution direction is exceptionally clear and easily understood by all stakeholders.
Modifiability	Solution is rigid and difficult to modify for future extensions.	Modification is possible but requires significant effort.	Solution allows for some modification but with considerable effort.	Solution is fairly easy to modify and adapt for future extensions.	Solution is designed with excellent foresight, making it very easy to extend and modify.
Maintainability	Code is poorly written with no adherence to coding standards; heavy reliance on case analysis and down-casts.	Some adherence to standards, but significant issues in code quality.	Code mostly adheres to standards with some reliance on case analysis and down-casts.	Code is well-written with minor issues; minimal reliance on case analysis and down-casts.	Code is exemplary, fully adheres to coding standards and avoids unnecessary complexities.
User Engagement	UI is poorly designed, not engaging, and hard to use.	UI is functional but lacks appeal and minimal engagement.	UI design is functional but lacks visual appeal or user engagement.	UI is visually appealing and engages users effectively.	UI is exceptionally designed, highly engaging, and enhances user experience.

Romal Design Review

Hirun Design Review

Referring to the Assessment Criteria outlined above, Alex's tech-based prototype has been assessed as the following:

1. Functional Completeness:
   - Needs Improvement(1): The implementation includes a baseline board setup visually, as there are 24 cards, 4 caves and 16 dragon cards present. And each dragon card position is randomised. The chosen functionality is implemented where each dragon card position is randomised and can be flipped over to reveal a creature. Cave creatures are randomised which is good. However, some improvements need to be made in regards to the ability to unflip the dragon cards, showing creatures on the squares of the volcano cards as well as the randomisation of the board.

2. Functional Appropriateness:
   - Very Good (3): The code logically follows the game's rules and objectives as detailed in the game manual. The solution's direction seems appropriate for simulating the board game environment, though some minor improvements could enhance the integration of game components.

3. Appropriateness Recognisability:
   - Very Good (3): The source code is structured in a way that it is easy to follow what class does what which makes it easy to understand where Hirun wants to go in terms of his solution direction. His class and method names are descriptive enough to understand their role for the game, making the codebase relatively easy to navigate.

4. Modifiability:
   - Very Good (3): The current code in regards to the flipping dragon cards game functionality is easy to modify as the solution is simple with few dependencies which makes it easy to for example add new types of cards or game features. The separation into different classes for handling specific game elements like cards and board setup supports extensibility. There could be some improvements to the modifiability by not hard coding the showing of the board

5. Maintainability:
   - Very Good (3): The code is well-organised, with clear separation of responsibilities of each class as well as good separation of responsibilities with the functions related to each class which facilitates maintenance. However, some sections could benefit from additional comments (file headers, function headers) to explain complex logic or decisions.

6. User Engagement (Aesthetics of the User Interface):

- Satisfactory (2): The user interface, based on the description of the `BoardPanel` class, utilises basic Java Swing components. The visuals are most definitely functional, the aesthetic appeal isn’t, and the interface could be enhanced to improve user engagement, such as more sophisticated graphics or interactive elements.

Summary:
The code effectively implements the fundamental mechanics of the "Fiery Dragons" game, adhering well to the game's rules and providing a functional user interface. While functionally complete and maintainable, there's room for enhancing the user interface and adding comprehensive documentation to boost understandability and ease of future modifications.

Alex Design Review

Referring to the Assessment Criteria outlined above, Alex’s tech-based prototype has been assessed as the following:

Object Oriented Design

CRC Cards

Class Diagram (UML)

Tech-based Software Prototype

Instructions for how to build and run the executable

Video Demonstration

There should be instructions on how to submit this video in the brief/edstem

For the demonstration we'll need to demonstrate these scenarios

- Using a number of game situations, demonstrate that your implementation correctly follows the rules of the game. You must illustrate at a minimum
    - an 'initial' board at the beginning of the game,
    - a player flips a Dragon card,
    - the dragon token moves correctly based on the last uncovered Dragon card,
    - a player uncovers another Dragon card if the turn is not over yet,
    - once a turn is over, flip all the Dragon cards face down for next player's turn,
    - a specific situation where a Dragon Pirate card is uncovered or a different animal (to the animal shown on the square where your dragon stands) is uncovered or your dragon would land on an occupied square, and
    - the end of the game.